Talk notes, presentation and demos are all in \\markbou001\public\Beta2samples\Talks\Intro:

WindowsForms.ppt – Latest presentation
Talk Notes.doc – notes on slides and demos and demo scripts – quality is variable
Enabling the rich client.doc - Draft white paper that serves as background for the no-touch deployment parts of the talk

Demos:

- **ChartDemo**
- **GDIPlusDemo**
- **NoTouch1**
- **NoTouch2**
- **ShapedWindowsDemo**

Video: http://mste/online/online.asp?id=001218netb altho' not one of my best attempts - I ramble too much

All demos are complete with the exception of the the ChartDemo is not completely ported to Beta2 – security stuff and Customization form has not yet been ported

Demo notes below.
Slide notes below

CustomerDetails Demo

Note: Needs updating for Beta2

Purpose of this demo is:

1. To show using VS.Net to build a Windows Forms app
2. To show anchoring and docking features
3. To show databinding to business objects (typically another talk will have shown databinding to ADO.Net)
4. To show that design time experience is very like that of VB6

Background:

Project consists of 2 projects:

CustomerList: A Class Library project that contains: CustomersList collection class, Customer object.
CustomerDetailsForm: A Windows App project that you use to build a Form that uses Customer & CustomersList.

Setup

Note: Once CustomerList.DLL is added to the CustomerDetailsForm project it is locked on disk and cannot build and so you need to
build CustomerList.DLL once to get the DLL and then turn off the build for this project.

Note: Once you've built this once, keep a copy of the exe and dll so that if all goes horribly wrong you can still show the running app

Create CustomerList.dll

- Open CustomerDetails.sln
- Right click on solution. Select Configuration Manager
- Check "Build" for both projects.
- Rebuild all
- Check that you have CustomerList.dll in CustomerList\bin
- Right click on solution. Select Configuration Manager
- Uncheck "Build" for CustomerList project.

Reset CustomerDetailsForm project:

Make sure that References **does not** contain a reference to CustomerList.dll
Make sure that Form1.vb is empty - delete Form1.vb and add a new Form1.vb (Add->New Item->Windows Form)
Make sure the CustomersList is not on the toolbox

Save the solution - you are now ready to go

Demo:

Before starting demo talk about UI on slide:

Listbox on left, splitter in middle, details view on right
ListBox contains a list of customers

When you select a customer in the list box details are shown on the right
Can use splitter to resize form just like windows explorer UI in

Start by demoing Forms designer, ambient properties, anchoring and docking, extensible property grid

- Open solution
- First thing we are going to do is to build our UI
- Double click on Form1.vb in solution explorer
- Resize form to about Size=500,400
- Going to start by adding the listbox
- Add a listbox to the form - size to about 170,200
- Select the Dock property in the property grid. Dock the ListBox to the left of the form
- Point out that the Dock property editor is a special editor that we have added for the Dock property. Point out that this mechanism is extensible. Component and control vendors can add their own editors directly into the property browser. Property browser is completely extensible. very cool.
- Show how list box is now docked to the left,top and bottom edges of the form by resizing the form in the designer
- Add a splitter by double-clicking on the splitter in the toolbox.
- Select form.
- Add panel to contain details view to rhs of form - size to about 250,200
- Set Panel1.BackColor to some foofy web color - let's say PowderBlue
- Set Panel1.Dock=Dock.Fill in the property browser
- Now add 2 labels, 2 text boxes, 2 buttons to the form:

```
Label1     TextBox1

Label2     TextBox2


   Button1 Button2
```

- Make sure that they are big enough to hold text once font in made bigger - next step
- Now make some comment about audience not being able to read form.
- Select Form by clicking on Form title bar in designer, Select Font in property browser
- Set font property to something like Tahoma 12pt Bold. Point out how all the controls on the form automatically pick up the new text size. Font, BackColor,ForeColor are automatically picked up by containing controls.
- Now set up controls:

  Label1: Text=Name
  Label2: Text=Address

  TextBox1:

  - Resize so that its left side is close to the label and right side is close to the form edge.
  - Text = ""
  - Anchor=Top,Left,Right.

  TextBox2:

- Resize so that its left side is close to the label and right side is close to the form edge.
- Text = ""
- Anchor=Top,Left,Right,Bottom
- Multiline=true
- Resize to about 194,144

Button1:

- Text=Load
- Anchor=Bottom,Right

Button2:

- Text=Save
- Anchor=Bottom,Right

- Select button1 and button2 and size so that they have the same size using the formatting toolbar along the top (demos formatting)
- Set button1 and button2 backcolor=(System)ControlDark.
- Set button1 and button2 FlatStyle=Flat - shows how to create flat style buttons

Talk about how anchoring works - anchors edges of control to edges of container. Show it in action at design time by resizing the form in the designer.

You are now done building the UI - save all and F5 to show UI and smart layout

Resize Form to show anchoring.
Resize splitter position to show docking and anchoring in tandem to give nice resizable forms with no code.
Close form

Next step - add some data

Say in this scenario we are going to display a collection of customer objects that could have been returned by a web service

Show Customer object in Customer.vb
Show CustomerList - standard collection. The things that make this something that can be used at design time are:

```
Public Class CustomersList
    Inherits CollectionBase
    Implements IComponent
    Implements IBindingList
```

ShowInToolBox attribute to get it to show up on the toolbox
Implements IComponent - in order to be designable a class must implement IComponent or derive from Component. Cannot derive from Component as it derives from the collection base class so implements IComponent - simple implementation that delegates to a contained component object:

```
Private compImpl As Component = New Component()
```

The final thing that makes it possible to set up databinding to this thing at design time is a typed (returns a Customer not object) Item property - in this case returns Customer:

```
Default Public Property Item(ByVal index As Integer) As Customer
   Get
      Return CType(List(index), Customer)
   End Get
   Set
      List(index) = value
   End Set
End Property
```

- So now we have our Customer Collection can simply add it to the toolbox.
- Go back to design view of Form
- Select general tab of toolbox
- Right click on toolbox. Select Customize toolbox...
- Select .Net framework components and Browse
- Navigate to CustomerList.DLL and add it to the list of components
- scroll down to CustomersList and select the check mark to add it to the toolbox. If you don't do this it won't be on the toolbox
- You should now have CustomersList on the toolbox.
- Drop it on the form - you get CustomersList1 on the component tray (this works because its a component)

Can now set up databinding

- Select the list box and set DataSource=CustomersList1 (you get 2 copies in the drop down - its a bug - select the first one). You can set DisplayMember to tell the Listbox what property to display (Customer.Name or Customer.Bar or whatever). If you don't set DisplayMember we call ToString() on the object. This works well in this case so we are not going to set DisplayMember.
- At this point you will start getting the nasty flickering wait/not wait mouse cursor. You can't ignore it. People will notice. Dismiss it with a joke "Its a beta1 feature so that you know how much work we are doing for you. Cut from Beta2 because usability studies showed you didn't need to know quite how busy we are. So enjoy this feature while you can". If you have a better joke please let me know <g>
- ListBox is now databound.
- Now bind the text boxes: Make the property browser wider. Select TextBox1. In the property browser select (Bindings). Expand it. Select Text. Drop down the list and select CustomersList1.FirstName. TextBox1 is now bound to the first name property of the current Customer in CustomersList1
- Repeat this for TextBox2.Text and CustomersList1.Address

Now add the code to load the data:

- Double click the Load button. Add the following code:

```
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
   CustomersList1.LoadCustomers()
End Sub
```

To load the customers and set the listbox to the first customer.

You are now done building the app - save all and F5 to show app running.

Hit load data button

Show details form changing as you select items in the listbox

Phew done!

I think that this is probably enough for a demo esp. for an hour talk. There are some bonus points:

- Add a menu to show cool menu editor
- Edit tab ordering to show cool tab ordering UI
- Add an error provider and show errors

NoTouch1 Demo

Setting up Simple Load:

1. Add a vroot to your default Web Server called SimpleFormsWeb - C:\Inetpub\wwwroot\SimpleFormsWeb

2. Create a "staging server" - say c:\Staging\SimpleFormsWeb

3. Open the solution

4. Set up build paths for dependent assemblies

   Select the SecondForm project.
   Open the Project properties.
   Select Configuration PRoperties.
   Select Build.
   Change Configuration in the drop down at the top to All Configurations
   Change Output path to c:\Staging\SimpleFormsWeb

   Select the FirstForm project.
   Open the Project properties.
   Select Configuration Properties.
   Select Build.
   Change Configuration in the drop down at the top to All Configurations
   Change Output path to c:\Staging\SimpleFormsWeb

5. Refresh Project Reference

   Select the FirstForm Project
   Select References
   Delete the reference to SecondForm
   Right click on References and Select Add
   Go to the Project Tab
   Doubleclick on SecondForm to add a project to project reference to SecondForm

6. Go register the Assembly viewer shell extension

   Open a cmd prompt
   cd \winnt\Microsoft.NET\Framework\v1.0.2204
   regsvr32 shfusion.dll

7. Add a favorites to the DownLoad cache

Open File Explorer
Navigate to \winnt\Assembly
Select Download (should be under Assembly)
Add as a favorite

Running SimpleLoad:

1. Make sure the Download cache is empty. Run "gacutil /cdl". I typically have a shortcut to this on my desktop so I can just double click on it.
NOTE: If you'd been running ASP.Net code you will need to stop and start IIS in order to clear the cache (IISRESET from the cmd line will do the trick).

2. Show 3 projects. Explain .exe and 2 dll assemblies. dll assemblies built to web server and are downloaded from web server.

3. Show MainForm Form and show Assembly.LoadFrom. Explain how this works.
LoadFrom loads the DLL from the web server across HTTP.
GetType get the type out of the assembly
Activator.CreateInstance like CreateObject in VB6.

4. Show FirstForm code that loads and shows Second Form. No special magic code here.
Because assemblies are self describing the runtime can understand that FirstFrom has a ref to SecondForm. When (and only when) you execute "new SecondForm" rhe runtime attempts to resolve that class. It looks for it first in the same
place as FirstFomr, and as a result downloads the SecondForm assembly from the web server at that point. Talk about trickle feed of app to the client etc. No need to download the whole app as a single enormous blob.
Fetch only what you need when its needed.

4. Build. Show SimpleForm and SecondForm assemblies on staging server. Copy them to the web site. Drag and drop deployment

5. Run the app (f5). Main Form comes up on the screen.

6. Show empty download cache. Nothing downloaded yet

7. Hit button. Form downlaods and shows. Show FirstForm in the download cache.

8. Hit OpenForm2 button on SimpleForm. Form downloads and shows. Show SecondForm in the download cache.

9. Kill App

10. Go to FirstForm in VS. Change the form back ground color or something equally visible. Rebuild. Repeat step 4-8 to show updates (remember to copy dlls to web server). When you show in the download cache. You'll have new versions of SimpleForm and SecondForm because by default VS updates version numbers on each build. Update is driven by version number or datetime of no version number.

(Step 11 is if offline comes up and small audience - big audience you'd have to spend too much time explaining this)

11. Go IE make offline. Go to SecondForm in VS. Change again. Repeat steps 4-8 to show no update on the client - you are working offline. Make online. Repeat steps 5-8 to show updates.

So now you have a model for building Smart Windows clients that have the same deployment characteristics as HTML pages in the browser: No need to install the whole app on the client.

You have to be able to afford to install the runtime on the client but once you've done that you can install a small stub onto the client and the rest of the app can be downloaded as required.
No need for massive install processes. Apps trickle feed onto the client. Just like HTML in the browser.

Updates to the web server automatically cause the client to be updated the next time the client app is started. Just like HTML in the browser.

You can take this even further. For instance if you have a web service that returns the list of assemblies for the client app to use you can tailor that list of assemblies so that each user gets a personalised client app that is suited to the task they are performing. Just like HTML in the browser. This is what NoTouch2 shows

NoTouch2 Demo

Setup and demo notes need to be written

Chart Demo Notes

Setup – this is Beta1 out of date needs updating

Setup VS

Set up Web Site

- Start VS
- New Project
- Visual Basic Projects
- New Web Application
- Call it ChartDemo or similar
- Delete the style sheet
- Delete the web form
- Add a folder called Common to the Web site
- Add a folder Control to the web site
- Right-click on Project->Add->existing item to add \\markbou001\public\Beta1Raw\WebSite\ChartDemo\default.htm to the Web site root folder
- Right-click on Common->Add->existing item to copy the files in \\markbou001\public\Beta1Raw\WebSite\ChartDemo\Common to the Common folder
- Right-click on Control->Add->existing item to copy the files in \\markbou001\public\Beta1Raw\WebSite\ChartDemo\Control to the Control folder

  You should end up with:

C:\inetpub\wwwroot\
ChartDemo <dir>
Default.html
Control <dir>
Default.html
*.dll

Common <dir>
pdc.css
pdcheaderlogo.bmp

- Select the **Control\default.html** in web project and right click and select "Set as Start Page"
- Save the solution

Set up Control Projects

ChartEngine

- Add a new C# Control Class Library project called ChartEngine to the solution
- Delete Control1.cs and Readme.txt
- Right-click on Project->Add->existing item to copy the files in \\markbou001\public\Beta1Raw\ChartDemo\ChartEngine to the project
- Right click on Project and open project properties
- Select Configuration Settings
- Select Build
- In the RHS select Output path and change it to the Control directory you created for the web project - C:\Inetpub\ wwwroot\ChartDemo\Control - this means the control will build to your web site directory
- Press the Apply Button
- Do this for both Release and Debug
- **Save all**

CustomizerForm

- Add a new VB Windows Control project called CustomizerForm to the solution
- Delete UserControl1.vb

- Right-click on Project->Add->existing item to copy the files in \\markbou001\public\Beta1Raw\ChartDemo\ CustomizerForm to the project
- Right click on Project and open project properties
- Select Configuration Settings
- Select Build
- Select Output path and change it to the Control directory you created for the web project - C:\Inetpub\wwwroot\ ChartDemo\Control - this means the control will build to your web site directory
- Press the Apply Button
- Do this for both Release and Debug
- Right-click on References->Add Reference
- Select the Projects Tab.
- Select ChartEngine
- Press Select and then OK
- Save All
- Build. You should not get build errors
- Save All

Charting

- Add a new C# Control Class Library project called Charting to the solution
- Delete Control1.cs and Readme.txt
- Right-click on Project->Add->existing item to copy the files in \\markbou001\public\Beta1Raw\ChartDemo\Charting to the project
- Right click on Project and open project properties

- Select Configuration Settings
- Select Build
- In the RHS select Output path and change it to the Control directory you created for the web project - C:\Inetpub\ wwwroot\ChartDemo\Control - this means the control will build to your web site directory
- Press the Apply Button
- Do this for both Release and Debug
- Save all
- Right-click on References->Add Reference
- Select the Projects Tab.
- Select ChartEngine and press Select
- Select CustomizerForm and press Select
- Press OK
- Build. You should not get build errors
- Save All

Back to the Solution

- Right-click on Solution->Configuration Manager. Set Active Solution Configuration to Release. This is important - the pdbs (debug files sometimes get locked and prevent rebuilding)
- Save All
- Build
- Go and look in C:\Inetpub\wwwroot\ChartDemo\Control - you should have the 3 DLLs - ChartEngine, Charting, CustomizerForm in there.

Say phew!

Running everything

Save All again just to be sure.

Hit Ctrl+F5 - ALWAYS USE CTRL+F5 not F5

The browser will start up and show the chart control - may be slow be patient!
(Note for browser control to show up you will need to change security settings – see demo script).

Final thing to do in VS is to add shortcuts to the code you want to show.

For instance the in style sheet pdc.css put the cursor on the line "background-color: beige;" and right click and select "Add Task List Short Cut".

Press Ctrl+Alt+Shift+S to bring up the task list (yes I said Ctrl+Alt+Shift+S). You'll
see an entry for the line in the pdc.css. You can rename this by left clicking and holding
the mouse down.

Add short cuts for:

- ChartPanel.cs(146): ChartCustomizerForm
- ChartPanel.cs(155): buttonLoadFile_Click
- Common\pdc.css(4): Style Sheet
- Control\default.html(43): Control Object Tag

Setup Security

You have to set up security so that the trusted zone has FullTrust

NOTE: Beta1 only – Beta2 this is not necessary

Open up a command window and type “caspol –lg”. You should see something like:

```
C:\>caspol -lg
Microsoft (R) COM+ caspol 2000.14.1812.10 - the policy
manipulation tool
Copyright (c) Microsoft Corp 1999-2000. All rights reserved.

Security is ON
Execution checking is OFF
```

```
Policy change prompt is ON

Level = Machine

Code Groups:

1.  All code: Nothing
   1.1.  Zone - MyComputer: FullTrust
      1.1.1.  Honor SkipVerification requests: SkipVerification
   1.2.  Zone - Intranet: LocalIntranet
   1.3.  Zone - Internet: Internet
   1.4.  Zone - Untrusted: Nothing
   1.5.  Zone - Trusted: Internet
   1.6.  StrongName -
00240000048000009400000006020000002400005253413100040000003
000000CFCB3291AA715FE99D40D49040336F9056D7886FED46775BC7BB5430BA44
44FEF8348EBD06
F962F39776AE4DC3B7B04A7FE6F49F25F740423EBF2C0B89698D8D08AC48D69CED
0FC8F83B465E08
07AC11EC1DCC7D054E807A43336DDE408A5393A48556123272CEEEE72F1660B719
27D38561AABF5C
AC1DF1734633C602F8F2D5: Everything
Success

C:\>
```

Now type **caspol -cg 1.5 FullTrust .** This changes the Trusted Zone to have permission to do anything.You should see something like:

```
C:\>caspol -cg 1.5 FullTrust
Microsoft (R) COM+ caspol 2000.14.1812.10 - the policy
manipulation tool
Copyright (c) Microsoft Corp 1999-2000. All rights reserved.

The operation you are performing will alter security policy.
Are you sure you want to perform this operation? (yes/no)
y
Changed code group permission set to "Everything" in Machine
level.
Success
```

Now type **caspol -lg** again**.**You should see something like:

```
C:\>caspol -lg
Microsoft (R) COM+ caspol 2000.14.1812.10 - the policy
manipulation tool
Copyright (c) Microsoft Corp 1999-2000. All rights reserved.

Security is ON
Execution checking is OFF
Policy change prompt is ON

Level = Machine

Code Groups:

1.  All code: Nothing
```

```
   1.1.  Zone - MyComputer: FullTrust
      1.1.1.  Honor SkipVerification requests: SkipVerification
   1.2.  Zone - Intranet: LocalIntranet
   1.3.  Zone - Internet: Internet
   1.4.  Zone - Untrusted: Nothing
   1.5.  Zone - Trusted: FullTrust
   1.6.  StrongName -
00240000048000009400000006020000002400005253413100040000003
000000CFCB3291AA715FE99D40D49040336F9056D7886FED46775BC7BB5430BA44
44FEF8348EBD06
F962F39776AE4DC3B7B04A7FE6F49F25F740423EBF2C0B89698D8D08AC48D69CED
0FC8F83B465E08
07AC11EC1DCC7D054E807A43336DDE408A5393A48556123272CEEEE72F1660B719
27D38561AABF5C
AC1DF1734633C602F8F2D5: Everything
Success
```

Now when you move the web site to the trusted zone the File Open dialog will work.

Setting up Fusion

- Go to c:\winnt\assembly and type **regsvr32 shfusion.dll.** This registers the fusion shell extension so that you can see fusion stored files in explorer
- Open Explorer. Go to c:\winnt\assembly. Expand the node in the LH treeview. You should see “Download”. Add this as a favorite in Favorites. (Right now you should see the Charting and ChartEngine in there.
- Copy \\markbou001\public\Beta1Raw\Storetst\storetst.exe to somewhere on your machine. This is a little fusion utility that you can use to clear the fusion cache by using the nd option – “storetst nd”.

Note well: Use “storetst nd” to clear the cache before starting your demo or you can’t demo things downloading. If you have been doing IIS/ASP.Net stuff you will need to reset IIS as well – use IISRESET.

Setting up Chart Host

This is a WinForms app that hosts the chart control.

- Start VS
- Create a new C# Windows Application project
- Delete Form1.cs and Readme.txt

- Right-click on Project->Add->existing item to copy the files in \\markbou001\public\Beta1Raw\ChartDemo\ChartHost to the project
- Add a reference to the ChartEngine and Charting DLLs
- Build
- Ctrl+F5 to run

Say phew that was simpler.

Note well: Make sure that you have this project closed when building the chart control or things will lock up. I typically just run the exe for this one.

I think that's everything.

Chart Demo script – this is Beta1 out of date needs updating

Control in the Browser Demo

Background

The Chart Control consists of the following:

ChartEngine Assembly (C#) – ChartEngine, ChartData, ChartLine classes
Charting Assembly (C#) – ChartControl, ChartPanel, ErrorPanel classes
CustomizerForm Assembly (VB) – CustomizerForm class

ChartData – the data for chart to be drawn – contains a collection of ChartLines
ChartLine – the data for one of the lines on the chart
ChartEngine – draws a chart based on the info in ChartData
ChartControl – a Win Forms control that draws a chart using the ChartEngine
ChartPanel – a Win Forms control that hosts the ChartControl and provides customization UI and the chart data file load/save
ErrorPanel – a custom error handler that displays errors in a big readable font for demo purposes

CustomizerForm – a form that allows customization of the chart by presenting the collection of ChartLines in the property browser – shows runtime use of the property browser!

The demo is setup as a solution in VS.Net. – 4 projects:

ChartDemo (Web Project) – the web site
ChartEngine (Control Class Library) **–** the chart engine
Charting (Control Class Library) **–** the chart controls
CustomizerForms (Control Class Library) **–** Customizer Form

Web Site is structured as follows:

C:\inetpub\wwwroot\
ChartDemo<dir>
Default.html
Control <dir>
Default.html
*.dll
Common <dir>
pdc.css
pdcheaderlogo.bmp

Startup Project: ChartDemo
Startup Page: Control\default.html

Build Process: The three DLL projects are setup to build retail into c:\inetpub\wwwroot\winformsdemos\control so that changing and rebuilding the controls updates the web site. Retail rather than debug because at one point I had pdb locking problems.

Security

Beta1: Trusted Zone has FullTrust because Windows Forms requires full trust to run in Beta1. **NOTE:** Beta1 only

Demo Synopsis

Control run in Local Machine App:

Purpose of this piece is to establish the control running as a trusted control on the local machine with full access to local resources. Sets stage for control in the browser.

- Show Chart Control in WinForms host.
- Show rich GDI+ based UI, resizing,

Control run in browser:

Shows:

- The same control running either locally or in the browser unchanged
- Automatic deployment of dependent assemblies + download on demand
- Browser integration – pick up font changes, style sheet changes
- Security – In Beta1 control can only run with full trust. In Beta2 will run with no trust but File IO will not be allowed.

- Show project in VS.

Deployment

- Show 3 assemblies.
- Show object tag – simply ref to Charting assembly and control.
- Show customize button event handler – all the code does is new the form.
- Show fusion assembly cache – empty.
- Hit Ctrl+F5 to run in the browser.
- Show assembly cache – ChartEngine automatically faulted in because referenced by the Charting assembly. Also note no ChartCustomizer assembly
- Go back to browser. Click on Customize button – displays cool UI for changing chart style (written in VB using property browser)
- Show assembly cache – CustomizerForm assembly has automatically been downloaded! All the code you wrote was “new CustomizerForm” and the RT/Fusion took care of the rest.

Property Browser at runtime

- Back to the browser – change a few things in the Customization dialog – show how cool being able to use the property browser is – form is less that 200 lines of VB code!!!!!

Browser Integration

- Back to browser – change font and show how control resizes. Leave as medium
- Back to VS change style sheet body.background-color -> pink
- Back to browser – F5 and it goes pink

Security

- When you first attempt to load chart control – fails to load because security engine stops it.
- Add web site to trusted zone and bingo the control works. Note well: Beta1 only limitation. In Beta2 will just work

Before starting

- Open command line – check security settings – caspol –lg should show “1.5. Zone – Trusted: FullTrust”. Close command line.
- Open IE – check View->Text Size is pointing to medium
- Open IE – check Tools->Internet Options->Security->Trusted Sites->Sites is empty
- Double Click on “Clear Cache” – clears the fusion cache
- Open Windows Explorer
- Use Favorites->Fusion – Download Cache
- Download Cache should be empty
- Minimise Explorer
- Double-Click on “Browser Demo” to start VS pointing to Controls in the Browser solution.
- Open pdc.css - make sure “body.background-color” is beige
- Minimise VS

Part 1 – Chart Control in Win Forms app

- Double-click on ChartHost
- Shows Chart Control running in a Win Forms app. Chart Data is Pig Iron Production from 1988 to 1992.
- GDI+ to draw – gradient fill etc.
- Resize and Form intelligently resizes (note demo so drawing not optimized)

Part 2 – Control in the Browser - deployment

- Go to open VS.
- Show WinFormsDemos solution in Solution Explorer
- Show Web Site and 3 projects – ChartEngine (C#) , Charting (VB) , CustomizerForm (C#). Each one produces an assembly. Charting ref's ChartEngine and CustomizerForm.
- Show Object Tag in Control\default.html – talk about classid – "URL/Assembly#Fully qualified class name"
- Go to Customize Button event handler in ChartPanel.cs – even tho' this is in another assembly written in VB can simply invoke (important later)
- Switch to Fusion Cache in Explorer – show that its empty
- Switch to VS
- Ctrl+F5 to run the solution **(MAKE SURE YOU USE Ctrl+F5 NOT F5!)**

Security

- Browser will fail to load Chart control – remember requires full trust.
- We have set up the permissions on this client so that web sites in the trusted zone have more permissions so we can add this site to the Trusted Zone and the control will be able to load files.
  - Go to Tools->Internet Options->Security->Trusted Sites->Sites. Add http://localhost. Hit OK/ Hit OK.
- Press F5 to refresh the browser

- Control will show in the browser. Its not fast. Be prepared!
- Switch to Fusion Cache in Explorer (may need to hit F5 to refresh) – show that Charting.DLL has downloaded and so has ChartEngine.DLL – even tho' nothing in the Web page for this – the RT has automatically downloaded ChartEngine.DLL because it is referenced and used. **Note** that CustomizerForm.DLL has not been downloaded because even tho' its ref'd it hasn't been used yet.
- Switch back to the browser. Click on the Customize button – we showed the code for this earlier. This displays cool UI for editing the ChartData.
  - First point: Switch to Fusion Cache in Explorer (may need to hit F5 to refresh) – show that CustomizerForm assembly has been downloaded on demand. No special code – RT just goes and finds the assembly when code says "new CustomizerForm"

- Second point: Edit Chart data using Property browser UI – change Soviet Union color to Gold and Width to 15. Written in VB. Less than 200 lines of code! On return to Chart note that Line is alpha blended (GDI+)

Browser Integration

- Switch to Browser
- Switch to VS
- Go to pdc.css style sheet. Change body.background-color to pink. Save.
- Switch to Browser – refresh the page (F5) . It goes pink. So does the control. Controls even pick up style sheet changes. Note limitations – background colors only not background images. **Note well Don't keep hitting F5. Refresh a couple of times and the control painting breaks.**

Setup VS

Set up Web Site

- Start VS
- New Project
- Visual Basic Projects
- New Web Application
- Call it ChartDemo or similar
- Delete the style sheet
- Delete the web form
- Add a folder called Common to the Web site
- Add a folder Control to the web site
- Right-click on Project->Add->existing item to add \\markbou001\public\Beta1Raw\WebSite\ChartDemo\default.htm to the Web site root folder
- Right-click on Common->Add->existing item to copy the files in \\markbou001\public\Beta1Raw\WebSite\ChartDemo\Common to the Common folder
- Right-click on Control->Add->existing item to copy the files in \\markbou001\public\Beta1Raw\WebSite\ChartDemo\Control to the Control folder

  You should end up with:

C:\inetpub\wwwroot\
ChartDemo <dir>
Default.html
Control <dir>
Default.html
*.dll
Common <dir>
pdc.css
pdcheaderlogo.bmp

Slide Notes

New term for apps that use client side processing power (ie Windows Forms apps): "The Smart Client" so Windows Forms apps are Smart Client apps.

Slide 1: What is it?

A framework for building Windows applications that takes advantage of all the power of .NET Framework and the Win32® API

Basic point is take advantage of the smart client - actually use all that client side processing power. Use the Win32 API so not slow and fat.

An architecture for controls and containers

Replacement for ActiveX without all the implementation issue mismatch of ActiveX because we supply the implementation of control/container communication

RAD design time experience - Integrated designer support
.NET Framework built with design-time in mind
Design-time built using .NET Framework: VS .NET Property Browser, Server Explorer, Windows Forms Designer

Designed desgin time and run time hand in hand rather than slapping design time on afterwards

We've dog fooded our own stuff. Proved that you can build real world shippable software with Windows Forms: Visual Studio.

Slide 2: Why do you care?

“No Touch” deployment

The big news is that we've fixed deployment and TCO.

Isolation is the default

Apps are isolated by default. Not forced to share unless you want to. Makes apps more stable. You can update isolated apps safe in the knowledge that only that app will be updated and it won't affect other apps on that machine.

Versionable - no side affects

You can still share components if you want to. We have a robust versioning mechanism for shared components so that apps bind to a specific version of a shared component and multiple versions of shared components can exist on the same machine at the same time - side by side - the app will continue to use the version it was built against. Again you can add new versions of shared components without affecting any of the installed apps. You can use config files to change this behavior - for instance if you need to apply a hot fix.

No registration - download and run

In order to run an app you don't need to register anything or install anything. You can simply copy an app to a machine and run it. If downlaod an app you can simply run it with no side effects and no registration.

Dependent components are downloaded as needed

If components are downloaded onto your machine the runtime chases references so that dependent components are downloaded as an when needed.

Smart caching and automatic updates

If you download components onto your machine the runtime will execute them

out of the client side cache until they are updated on the server and then it will download the new ones as they are needed.

Secure
Can run as semi-trusted browser component
The code access security engine in the runtime makes running controls downloaded from web sites safe. Unlike ActiveX the CAS engine is pro-active and won't allow bad code to run on your machine.

Full integration with .NET Framework
Web Service aware
Easy to build Windows Forms apps that are web service aware using VS.Net. Just rag and drop web services.....

Data aware
Easy to build Windows Forms apps that take advantage of all of the features of ADO.Net. Databinding architecture allows you to plug business objects into databinding.

Slide 3: Security

Code access security makes
Windows Forms controls safe
The code access security engine in the runtime makes running controls downloaded from web sites safe. Unlike ActiveX the CAS engine is pro-active and won't allow bad code to run on your machine.

Execution controlled by:
Permissions
Based on permissions: Permission to Access a File, Permission to display a Window, Permission to read the registry.

Permission Sets
Group permissions together into Permission Set. A Permission set might say: "You can display top level windows and controls and you can talk to a web service back to your originating web server, but you cannot access the clipboard or the file system or the registry".

Code Groups (Zones)
Code Groups are about where code comes from (the evidence for that code). The easiest to understand code groups are the zones in IE. But code groups are very flexible. For instance you can define a code group based on the digital signature of an assembly. This means that you can sign all your code with your private key and give that code more trust so that everything you write has more trust that everything you buy from a third party.

Code groups have permission sets
You associate permission sets with code groups. So you might say all code downloaded from the internet has no permissions at all and all code downloaded from the intranet has the permission set we talked about before: "You can display top level windows and controls and you can talk to a web service back to your originating web server, but you cannot access the clipboard or the file system or the registry".

Then whenever code is downloaded from the internet it won't be able to run at all. The runtime will not let it run.

When code is downloaded from the intranet it will not be able to read and write to the registry. The runtime won't let it. If code downloaded from the intranet trys to access the registry the runtime will throw a security exception.

This makes Windows Forms controls running in the browser safe. This makes Windows Form apps running from a web server safe.

Permissions sets and code groups are controlled by the systems admin. not the end user. This is very important. No longer forcing the end user (least qualified person) to make security decisions. Moved back into the control of the systems and security admins for an org.

Means no need for the nasty dialog in IE that no-one understands because security is determined by the runtime based on the config defined by the admin not by the end user.

No big switch - all or nothing - so not like activex. Admin can control what code does what.

2 tools: caspol - cmd line tool that can be scripted. MMC plugin that can be used to create security policy.